In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
data = pd.read_csv("Bengaluru_House_Data.csv")
```

In [3]:
```python
data.head()
```

Out[3]:

	area_type	availability	location	size	society	total_sqft	bath	balcony	price
0	Super built-up Area	19-Dec	Electronic City Phase II	2 BHK	Coomee	1056	2.0	1.0	39.07
1	Plot Area	Ready To Move	Chikka Tirupathi	4 Bedroom	Theanmp	2600	5.0	3.0	120.00
2	Built-up Area	Ready To Move	Uttarahalli	3 BHK	NaN	1440	2.0	3.0	62.00
3	Super built-up Area	Ready To Move	Lingadheeranahalli	3 BHK	Soiewre	1521	3.0	1.0	95.00
4	Super built-up Area	Ready To Move	Kothanur	2 BHK	NaN	1200	2.0	1.0	51.00

In [4]:
```python
data.shape
```

Out[4]:
```
(13320, 9)
```

In [5]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   area_type     13320 non-null  object
 1   availability  13320 non-null  object
 2   location      13319 non-null  object
 3   size          13304 non-null  object
 4   society       7818 non-null   object
 5   total_sqft    13320 non-null  object
 6   bath          13247 non-null  float64
 7   balcony       12711 non-null  float64
 8   price         13320 non-null  float64
dtypes: float64(3), object(6)
memory usage: 624.4+ KB
```

In [6]:
```python
#Value counts for each columns

for column in data.columns:
    print(data[column].value_counts())
    print("*"*20)
```

```
Super built-up  Area    8790
Built-up  Area          2418
Plot  Area              2025
Carpet  Area              87
Name: area_type, dtype: int64
********************
Ready To Move    10581
18-Dec             307
18-May             295
18-Apr             271
18-Aug             200
                 ...
15-Jun               1
16-Jan               1
16-Oct               1
16-Nov               1
14-Jul               1
Name: availability, Length: 81, dtype: int64
********************
Whitefield                540
Sarjapur  Road            399
Electronic City           302
Kanakpura Road            273
Thanisandra               234
                         ...
Near ullas theater          1
Near Electronic City,       1
kg halli jalhalli west      1
sarjapura main road         1
Jagadish Nagar              1
Name: location, Length: 1305, dtype: int64
********************
2 BHK         5199
3 BHK         4310
4 Bedroom      826
4 BHK          591
3 Bedroom      547
1 BHK          538
2 Bedroom      329
5 Bedroom      297
6 Bedroom      191
1 Bedroom      105
8 Bedroom       84
7 Bedroom       83
5 BHK           59
9 Bedroom       46
6 BHK           30
7 BHK           17
1 RK            13
10 Bedroom      12
9 BHK            8
8 BHK            5
10 BHK           2
11 BHK           2
11 Bedroom       2
13 BHK           1
27 BHK           1
14 BHK           1
19 BHK           1
16 BHK           1
12 Bedroom       1
18 Bedroom       1
43 Bedroom       1
Name: size, dtype: int64
********************
GrrvaGr    80
PrarePa    76
Sryalan    59
Prtates    59
GMown E    56
           ..
VVbhion     1
Pae 2ov     1
Adace P     1
Varlsa      1
Mebhaee     1
Name: society, Length: 2688, dtype: int64
********************
1200           843
1100           221
1500           205
2400           196
600            180
              ...
3124             1
42000            1
5665.84          1
673              1
4400 - 6640      1
Name: total_sqft, Length: 2117, dtype: int64
********************
2.0     6908
3.0     3286
4.0     1226
1.0      788
5.0      524
6.0      273
7.0      102
8.0       64
9.0       43
10.0      13
12.0       7
13.0       3
11.0       3
16.0       2
27.0       1
18.0       1
40.0       1
15.0       1
14.0       1
Name: bath, dtype: int64
********************
2.0    5113
1.0    4897
3.0    1672
0.0    1029
Name: balcony, dtype: int64
********************
75.00    310
65.00    302
55.00    275
60.00    270
45.00    240
        ...
81.55      1
69.49      1
42.18      1
70.25      1
74.82      1
Name: price, Length: 1994, dtype: int64
********************
```

In [7]:
```python
data.isnull().sum()
```

Out[7]:
```
area_type          0
availability       0
location           1
size              16
society         5502
total_sqft         0
bath              73
balcony          609
price              0
dtype: int64
```

In [8]:
```python
data.drop(columns = ['area_type','availability','society','balcony'], inplace = True)
```

In [9]:
```python
data.describe()
```

Out[9]:

	bath	price
count	13247.000000	13320.000000
mean	2.692610	112.565627
std	1.341458	148.971674
min	1.000000	8.000000
25%	2.000000	50.000000
50%	2.000000	72.000000
75%	3.000000	120.000000
max	40.000000	3600.000000

In [10]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   location    13319 non-null  object
 1   size        13304 non-null  object
 2   total_sqft  13320 non-null  object
 3   bath        13247 non-null  float64
 4   price       13320 non-null  float64
dtypes: float64(2), object(3)
memory usage: 364.3+ KB
```

In [11]:
```python
data['location'].value_counts()
```

Out[11]:
```
Whitefield                540
Sarjapur  Road            399
Electronic City           302
Kanakpura Road            273
Thanisandra               234
                         ...
Near ullas theater          1
Near Electronic City,       1
kg halli jalhalli west      1
sarjapura main road         1
Jagadish Nagar              1
Name: location, Length: 1305, dtype: int64
```

In [12]:
```python
#There is only one missing value in location so we are filling it with Sarjapur Road
data['location'] = data['location'].fillna("Sarjapur Road")
```

In [13]:
```python
data['size'].value_counts()
```

Out[13]:
```
2 BHK         5199
3 BHK         4310
4 Bedroom      826
4 BHK          591
3 Bedroom      547
1 BHK          538
2 Bedroom      329
5 Bedroom      297
6 Bedroom      191
1 Bedroom      105
8 Bedroom       84
7 Bedroom       83
5 BHK           59
9 Bedroom       46
6 BHK           30
7 BHK           17
1 RK            13
10 Bedroom      12
9 BHK            8
8 BHK            5
10 BHK           2
11 BHK           2
11 Bedroom       2
13 BHK           1
27 BHK           1
14 BHK           1
19 BHK           1
16 BHK           1
12 Bedroom       1
18 Bedroom       1
43 Bedroom       1
Name: size, dtype: int64
```

In [14]:
```python
#There are 16 null values in size hence we are filling those values with 2BHK
data['size'] = data['size'].fillna("2 BHK")
```

In [15]:
```python
#We have 73 missing values in bathroom so we are placing values with median
data['bath'] = data['bath'].fillna(data['bath'].median())
```

In [16]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13320 entries, 0 to 13319
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   location    13320 non-null  object
 1   size        13320 non-null  object
 2   total_sqft  13320 non-null  object
 3   bath        13320 non-null  float64
 4   price       13320 non-null  float64
dtypes: float64(2), object(3)
memory usage: 364.3+ KB
```

In [17]:
```python
data['bhk'] = data['size'].str.split().str.get(0).astype(int)
```

In [18]:
```python
data[data.bhk > 20]
#There are 2 location where we have more than 27 BHK, So these are outliers in data so we need to fix it.
```

Out[18]:

	location	size	total_sqft	bath	price	bhk
1718	2Electronic City Phase II	27 BHK	8000	27.0	230.0	27
4684	Munnekollal	43 Bedroom	2400	40.0	660.0	43

In [19]:
```python
data['total_sqft'].unique()
#We have got the values in form of ranges we need to fix it so we will take mean of ot like(1133+1384)/2
```

Out[19]:
```
array(['1056', '2600', '1440', ..., '1133 - 1384', '774', '4689'],
      dtype=object)
```

In [20]:
```python
def convertRange(x):

    temp = x.split(",")
    if len(temp) == 2:
        return (float(temp[0]) +float(temp[1]))/2
    try:
        return float(x)
    except:
        return None
```

In [21]:
```python
data['total_sqft']= data['total_sqft'].apply(convertRange)
```

In [22]:
```python
data.head()
```

Out[22]:

	location	size	total_sqft	bath	price	bhk
0	Electronic City Phase II	2 BHK	1056.0	2.0	39.07	2
1	Chikka Tirupathi	4 Bedroom	2600.0	5.0	120.00	4
2	Uttarahalli	3 BHK	1440.0	2.0	62.00	3
3	Lingadheeranahalli	3 BHK	1521.0	3.0	95.00	3
4	Kothanur	2 BHK	1200.0	2.0	51.00	2

price per square feet

In [23]:
```python
#Price/total_sqft=price per square ft

data['price_per_sqft'] = data['price']*100000/data['total_sqft']
```

In [24]:
```python
data['price_per_sqft']
```

Out[24]:
```
0         3699.810606
1         4615.384615
2         4305.555556
3         6245.890861
4         4250.000000
             ...
13315     6689.834926
13316    11111.111111
13317     5258.545136
13318    10407.336319
13319     3090.909091
Name: price_per_sqft, Length: 13320, dtype: float64
```

In [25]:
```python
data.describe()
```

Out[25]:

	total_sqft	bath	price	bhk	price_per_sqft
count	13073.000000	13320.000000	13320.000000	13320.000000	1.307300e+04
mean	1554.942029	2.688814	112.565627	2.802778	7.949600e+03
std	1238.458773	1.338754	148.971674	1.294496	1.072440e+05
min	1.000000	1.000000	8.000000	1.000000	2.678298e+02
25%	1100.000000	2.000000	50.000000	2.000000	4.265734e+03
50%	1275.000000	2.000000	72.000000	3.000000	5.454545e+03
75%	1670.000000	3.000000	120.000000	3.000000	7.338057e+03
max	52272.000000	40.000000	3600.000000	43.000000	1.200000e+07

In [26]:
```python
data['location'].value_counts()
```

Out[26]:
```
Whitefield         540
Sarjapur  Road     399
Electronic City    302
Kanakpura Road     273
Thanisandra        234
                  ...
Chokkasandra         1
Rahmath Nagar        1
 Whitefield          1
Sonam Layout         1
Escorts Colony       1
Name: location, Length: 1306, dtype: int64
```

In [27]:
```python
#Any location having < 10 value_counts replace the location with others

data['location'] = data['location'].apply(lambda x: x.strip())
location_count = data['location'].value_counts()
```

In [28]:
```python
location_count_less_10 = location_count[location_count<=10]
location_count_less_10
#This are the location having <10 counts
```

Out[28]:
```
Sector 1 HSR Layout      10
1st Block Koramangala    10
Sadashiva Nagar          10
Naganathapura            10
Nagadevanahalli          10
                         ..
Chokkasandra              1
Rahmath Nagar             1
Sonam Layout              1
Dwaraka Nagar             1
Escorts Colony            1
Name: location, Length: 1054, dtype: int64
```

In [29]:
```python
data['location'] = data['location'].apply(lambda x: 'other' if x in location_count_less_10 else x)
```

In [30]:
```python
data['location'].value_counts()
#So we got 2886 other values which was having <10 vlaue_counts
```

Out[30]:
```
other                          2886
Whitefield                      541
Sarjapur  Road                  399
Electronic City                 304
Kanakpura Road                  273
                               ...
Tindlu                           11
Nehru Nagar                      11
Pattandur Agrahara               11
2nd Phase Judicial Layout        11
Thyagaraja Nagar                 11
Name: location, Length: 242, dtype: int64
```

Outlier Detection and removal

In [31]:
```python
data.describe()
```

Out[31]:

	total_sqft	bath	price	bhk	price_per_sqft
count	13073.000000	13320.000000	13320.000000	13320.000000	1.307300e+04
mean	1554.942029	2.688814	112.565627	2.802778	7.949600e+03
std	1238.458773	1.338754	148.971674	1.294496	1.072440e+05
min	1.000000	1.000000	8.000000	1.000000	2.678298e+02
25%	1100.000000	2.000000	50.000000	2.000000	4.265734e+03
50%	1275.000000	2.000000	72.000000	3.000000	5.454545e+03
75%	1670.000000	3.000000	120.000000	3.000000	7.338057e+03
max	52272.000000	40.000000	3600.000000	43.000000	1.200000e+07

In [32]:
```python
#data whose values sqft/bhk <300 that should not be an feeseble flat

data = data[((data['total_sqft']/data['bhk'])>=300)]
data.describe()
```

Out[32]:

	total_sqft	bath	price	bhk	price_per_sqft
count	12329.000000	12329.000000	12329.000000	12329.000000	12329.000000
mean	1590.166773	2.561441	111.444236	2.651472	6322.476758
std	1261.827604	1.072551	152.759322	0.973754	4187.479096
min	300.000000	1.000000	8.440000	1.000000	267.829813
25%	1118.000000	2.000000	49.340000	2.000000	4207.119741
50%	1300.000000	2.000000	70.000000	3.000000	5300.000000
75%	1700.000000	3.000000	115.000000	3.000000	6938.483548
max	52272.000000	16.000000	3600.000000	16.000000	176470.588235

In [33]:
```python
data.shape
```

Out[33]:
```
(12329, 7)
```

In [34]:
```python
data.price_per_sqft.describe()
```

Out[34]:
```
count     12329.000000
mean       6322.476758
std        4187.479096
min         267.829813
25%        4207.119741
50%        5300.000000
75%        6938.483548
max      176470.588235
Name: price_per_sqft, dtype: float64
```

In [35]:
```python
def remove_outliers_sqft(df):
    df_output = pd.DataFrame()
    for key,subdf in df.groupby('location'):
        m = np.mean(subdf.price_per_sqft)

        st = np.std(subdf.price_per_sqft)

        gen_df = subdf[(subdf.price_per_sqft > (m-st)) & (subdf.price_per_sqft <= (m+st))]
        df_output = pd.concat([df_output,gen_df],ignore_index=True)
    return df_output
data = remove_outliers_sqft(data)
data.describe()
```

Out[35]:

	total_sqft	bath	price	bhk	price_per_sqft
count	10153.000000	10153.000000	10153.000000	10153.000000	10153.000000
mean	1502.725758	2.471880	91.104765	2.573525	5669.621675
std	873.997461	0.971938	86.241338	0.893533	2274.739750
min	300.000000	1.000000	10.000000	1.000000	1250.000000
25%	1110.000000	2.000000	49.000000	2.000000	4242.424242
50%	1283.000000	2.000000	67.000000	2.000000	5183.823529
75%	1650.000000	3.000000	100.000000	3.000000	6451.612903
max	30400.000000	16.000000	2200.000000	16.000000	24509.803922

In [36]:
```python
def bhk_outlier_remover(df):
    exclude_indices = np.array([])
    for location, location_df in df.groupby('location'):
        bhk_stats = {}
        for bhk, bhk_df in location_df.groupby('bhk'):
            bhk_stats[bhk] = {
                'mean':np.mean(bhk_df.price_per_sqft),
                'std':np.std(bhk_df.price_per_sqft),
                'count':bhk_df.shape[0]
            }
        for bhk, bhk_df in location_df.groupby('bhk'):
            stats = bhk_stats.get(bhk-1)
            if stats and stats['count']>5:
                exclude_indices = np.append(exclude_indices, bhk_df[bhk_df.price_per_sqft<(stats['mean'])].index.values)
    return df.drop(exclude_indices,axis='index')
```

In [37]:
```python
data = bhk_outlier_remover(data)
```

In [38]:
```python
data.shape
```

Out[38]:
```
(7227, 7)
```

In [39]:
```python
data
#This is our clean data.
```

Out[39]:

	location	size	total_sqft	bath	price	bhk	price_per_sqft
0	1st Block Jayanagar	4 BHK	2850.0	4.0	428.0	4	15017.543860
1	1st Block Jayanagar	3 BHK	1630.0	3.0	194.0	3	11901.840491
2	1st Block Jayanagar	3 BHK	1875.0	2.0	235.0	3	12533.333333
3	1st Block Jayanagar	3 BHK	1200.0	2.0	130.0	3	10833.333333
4	1st Block Jayanagar	2 BHK	1235.0	2.0	148.0	2	11983.805668
...	...	...	...	...	...	...	...
10144	other	2 BHK	1200.0	2.0	70.0	2	5833.333333
10145	other	1 BHK	1800.0	1.0	200.0	1	11111.111111
10148	other	2 BHK	1353.0	2.0	110.0	2	8130.081301
10149	other	1 Bedroom	812.0	1.0	26.0	1	3201.970443
10152	other	4 BHK	3600.0	5.0	400.0	4	11111.111111

7227 rows × 7 columns

In [40]:
```python
data.drop(columns=['size','price_per_sqft'],inplace=True)
```

Cleaned Data

In [41]:
```python
data.head()
```

Out[41]:

	location	total_sqft	bath	price	bhk
0	1st Block Jayanagar	2850.0	4.0	428.0	4
1	1st Block Jayanagar	1630.0	3.0	194.0	3
2	1st Block Jayanagar	1875.0	2.0	235.0	3
3	1st Block Jayanagar	1200.0	2.0	130.0	3
4	1st Block Jayanagar	1235.0	2.0	148.0	2

In [42]:
```python
data.to_csv('Cleaned_data.csv')
```

In [43]:
```python
X = data.drop(columns=['price'])
y = data['price']
```

In [44]:
```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score
```

In [45]:
```python
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,random_state=0)
```

In [46]:
```python
print(X_train.shape)
print(y_test.shape)
```

```
(5781, 4)
(1446,)
```

Applying Linear Regression

In [47]:
```python
column_trans = make_column_transformer((OneHotEncoder(sparse=False),['location']),
                                       remainder='passthrough')
```

In [48]:
```python
scaler = StandardScaler()
```

In [49]:
```python
lr = LinearRegression(normalize=True)
```

In [50]:
```python
pipe = make_pipeline(column_trans, scaler, lr)
```

In [51]:
```python
pipe.fit(X_train, y_train)
```

Out[51]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(sparse=False),
                                                  ['location'])])),
                ('standardscaler', StandardScaler()),
                ('linearregression', LinearRegression(normalize=True))])
```

In [52]:
```python
y_pred_lr = pipe.predict(X_test)
```

In [53]:
```python
r2_score(y_test, y_pred_lr)
```

Out[53]:
```
0.8155491448551294
```

Applying Lasso

In [54]:
```python
lasso = Lasso()
```

In [55]:
```python
pipe = make_pipeline(column_trans,scaler, lasso)
```

In [56]:
```python
pipe.fit(X_train, y_train)
```

Out[56]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(sparse=False),
                                                  ['location'])])),
                ('standardscaler', StandardScaler()), ('lasso', Lasso())])
```

In [57]:
```python
y_pred_lasso = pipe.predict(X_test)
r2_score(y_test, y_pred_lasso)
```

Out[57]:
```
0.8019408594870728
```

Applying Ridge

In [58]:
```python
ridge = Ridge()
```

In [59]:
```python
pipe = make_pipeline(column_trans, scaler, ridge)
```

In [60]:
```python
pipe.fit(X_train, y_train)
```

Out[60]:
```
Pipeline(steps=[('columntransformer',
                 ColumnTransformer(remainder='passthrough',
                                   transformers=[('onehotencoder',
                                                  OneHotEncoder(sparse=False),
                                                  ['location'])])),
                ('standardscaler', StandardScaler()), ('ridge', Ridge())])
```

In [61]:
```python
y_pred_ridge = pipe.predict(X_test)
r2_score(y_test, y_pred_ridge)
```

Out[61]:
```
0.8153748464584635
```

Applying all the three

In [62]:
```python
print("No Regularization: ", r2_score(y_test, y_pred_lr))
print("Lasso: ", r2_score(y_test, y_pred_lasso))
print("Ridge: ", r2_score(y_test, y_pred_ridge))
```

```
No Regularization:  0.8155491448551294
Lasso:  0.8019408594870728
Ridge:  0.8153748464584635
```

In [63]:
```python
import pickle
```

In [64]:
```python
#We are getting Ridge and Linear with almost same results so we have to dump any of it,in this case dump (Ridge)

pickle.dump(pipe, open("RidgeModel.pkl", "wb"))
```

In [ ]: